Jan 28th

Tuesday, January 28, 2014 10:01 AM

Pstree shows you in the tree format what the relationships are between programs running

What happens when the relationship is broken.
If the child dies first, then the process goes into a "Zombie" state
Every process has a PCB, when the process dies the PCB does not immediately get deleted, it stays into terminate state.
It will exists until another process checks the PCB. Wait(pid), this returns the value of the PCB and the PCB then is removed.

If the parent exits before the child, then the PCB gets reassigned to the "Bash Shell"

The wait command clears up the zombie state, see nano fork1.c

In c everything is passed by copy. Thus I need the & because that is the reference to the value.

On the other hand if the parent dies before the child, then that is called the orphan state.
Who then is responsible, the INIT state cleans up the process.
INIT then waits.

Inter Process Communication (IPC)
When communicating between multiple processes

When we need more communication between child and parent

1. Shared Memory
    a. This has to be allocated, otherwise the memory is separate
    b. This is where both process request memory, they both have memory added, that added memory points to the same memory block on the stack.
    c. It is up to the processes to make sure that there is a form of synchronization
2. The other way is called Message Passing
    a. A mediated way for them to communicate
    b. The most common way is pipes / FIFO in linux
    c. | this character is the pipe character
        i. In bash whenever a program is opened three files are opened
            1) Stdin
            2) Stdout
            3) Stderr
        ii. This | pipe command creates a link between the stdout and stdin files of the of the two programs
            1) Example wc(word count), it waits until the eof or end of file char which is ctrl d
            2) The cat functions will put all the information in a pipe. Pipe means stdout to stdin
                a) Example cat fork.c | wc
    d. These are called file streams, stdin and stdout
    e. You can create these streams in linux by calling mkfifo
        i. This file can be read from and written to, this is considered a string file, which means

other processes can open read to and write from etc…
   ii. This allows you to create your own named pipe.
f. This type relies heavily on the OS, all we are doing is calling std I/O functions.

Remote procedure Calls (RPC)

1. This is done to not only call the function but what process should be calling that function
   a. The OS has a table of all functions
   b. When the procedure is called, it looks it up in the table, goes into the process location usually the child and returns it to the parent.
   c. These are convenient but tricky and complicated

Sockets

1. Usually done between different machines
   a. When a linux machine is running there is a list of sockets available to an OS (65,000) each machine has it's own sockets. This is dependent on the number of network adapters available.
   b. What happens is when process on a machine is running it will listen to activity on a certain Port or Socket
   c. Then another machine comes along and connects to that machine on the named port.
   d. This creates a serial connection between the two,
   e. The process in machine two is put on the ready queue and runs or reads
   f. Machine1 will then write to machine 2